

# TCSS 422: OPERATING SYSTEMS


## Beyond Physical Memory, I/O Devices, Hard Disk Drives,

Wes J. Lloyd  
School of Engineering and Technology  
University of Washington - Tacoma

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington

Tacoma



1

## OBJECTIVES – 6/3

- **Questions from 5/29**
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.2

2

ONLINE DAILY FEEDBACK SURVEY

■ Daily Feedback Quiz in Canvas – Available After Each Class

■ Extra credit available for completing surveys **ON TIME**

■ Tuesday surveys: due by ~ Wed @ 11:59p

■ Thursday surveys: due ~ Mon @ 11:59p

TCSS 422 A > Assignments

Spring 2021

Search for Assignment

Home

Announcements

Zoom

Syllabus

Assignments

Discussions

Upcoming Assignments

TCSS 422 - Online Daily Feedback Survey - 4/1

Available until Apr 5 at 11:59pm | Due Apr 5 at 10pm | -/1 pts

June 3, 2025

TCSS422: Computer Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.3

3

TCSS 422 - Online Daily Feedback Survey - 4/1

Quiz Instructions

Question 1

0.5 pts

On a scale of 1 to 10, please classify your perspective on material covered in today's class:

12345678910

Mostly Review To MeEqual New and ReviewMostly New to Me

Question 2

0.5 pts

Please rate the pace of today's class:

12345678910

SlowJust RightFast

June 3, 2025

TCSS422: Computer Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.4

4

Slides by Wes J. Lloyd

L18.2

## MATERIAL / PACE

- Please classify your perspective on material covered in today’s class (45 of 63 respondents – 71.4%):
  - 1-mostly review, 5-equal new/review, 10-mostly new
  - **Average – 5.13 (↓ - previous 6.00)**
- Please rate the pace of today’s class:
  - 1-slow, 5-just right, 10-fast
  - **Average – 5.00 (↓ - previous 5.28)**

June 3, 2025	TCSS422: Computer Operating Systems [Spring 2025] School of Engineering and Technology, University of Washington - Tacoma	L18.5
--------------	--	-------

5


## FEEDBACK FROM 5/29

June 3, 2025	TCSS422: Operating Systems [Spring 2025] School of Engineering and Technology, University of Washington - Tacoma	L18.6
--------------	---	-------

6

# QUIZ 2 RESULTS

- The quarter does not end after the midterm



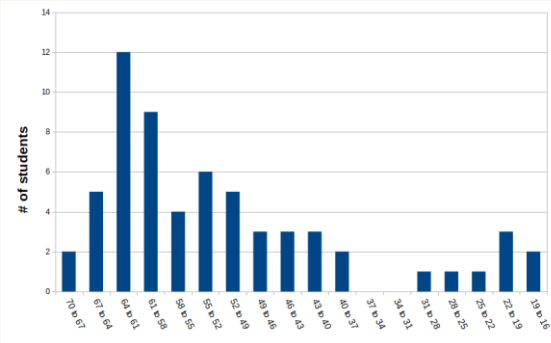
Next

June 3, 2025	TCSS422: Operating Systems [Spring 2025] School of Engineering and Technology, University of Washington - Tacoma	L18.7
--------------	---	-------

7

# QUIZ 2 RESULTS

- Statistics
- Average: 51.16
- Mode: 58.0
- Median: 55.5
- Min score: 16
- Lower quartile: 44.75
- 2<sup>nd</sup> quartile: 55.5
- 3<sup>rd</sup> quartile: 61
- Max score: 68
- Standard deviation: 13.74
- Curve: +9 pts, → brings class avg to=85.94%
- Question 6 Correction: +1



TCSS 422A Spring 2025 Quiz 2

May 13, 2025	TCSS562: Software Engineering for Cloud Computing [Spring 2025] School of Engineering and Technology, University of Washington - Tacoma	L12.8
--------------	--	-------

8

OBJECTIVES – 6/3

- Questions from 5/29
- **Assignment 2 - June 5 AOE**
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.9

9

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- **Assignment 3 (as a Tutorial) - June 10 AOE**
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.10

10

## TUTORIAL - ASSIGNMENT 3: INTRODUCTION TO LINUX KERNEL MODULES

- Assignment 3 provides an introduction to kernel programming by demonstrating how to create a Linux Kernel Module as a tutorial
- Kernel modules are commonly used to write device drivers and can access protected operating system data structures
  - For example: Linux `task_struct` process data structure

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.11

11

## OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- **Memory Segmentation Activity + answers (available in Canvas)**
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.12

12

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.13

13

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.14

14

FINAL EXAM – THURSDAY JUNE 12 @  
3:40PM<sup>TH</sup>

- Thursday June 12 from 3:40 to 5:40 pm
  - Final (100 points)
  - SHORT: similar number of questions as the midterm
  - 2-hours
  - Focus on new content - since the midterm (~70% new, 30% before)
- Final Exam Review -
  - Complete Memory Segmentation Activity
  - Complete Quiz 4
  - Practice Final Exam Questions – 2<sup>nd</sup> hour of June 5<sup>th</sup> class session
  - Quiz 2 Review – June 5<sup>th</sup> class session
  - Individual work
  - 3 pages of notes (any sized paper), double sided
  - Basic calculators allowed
  - NO smartphones, laptop, book, Internet, group work

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.15

15

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- **Tutorial 3 - File Systems (Optional, Extra Credit)**
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.16

16

CATCH UP FROM LECTURE 17

- Switch to Lecture 17 Slides
- Slides L17.32 to L17.80
  - Chapter 19 – Translation Lookaside Buffer
  - Chapter 20 – Paging Smaller Tables

May 29, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L17.17

17

WE WILL RETURN AT  
5:07PM

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma



L18.18

18

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- **Chapter 21/22: Beyond Physical Memory**
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives


June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.19

19

CHAPTER 21/22:  
BEYOND PHYSICAL  
MEMORY



June 3, 2025

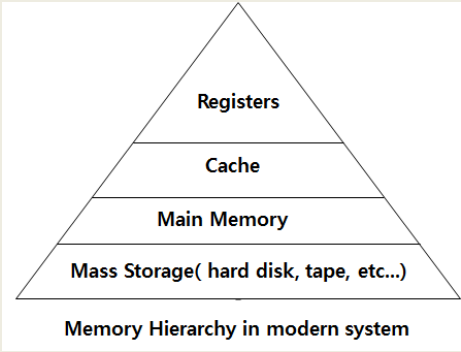
TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.20

20

## MEMORY HIERARCHY

- Disks (HDD, SSD) provide another level of storage in the memory hierarchy



Memory Hierarchy in modern system

June 3, 2025	TCSS422: Operating Systems [Spring 2025] School of Engineering and Technology, University of Washington - Tacoma	L18.21
--------------	---	--------

21

## MOTIVATION FOR EXPANDING THE ADDRESS SPACE

- Provide the illusion of an address space larger than physical RAM
- For a single process
  - Convenience
  - Ease of use
- For multiple processes
  - Large virtual memory space supports running *many concurrent processes. . .*

June 3, 2025	TCSS422: Operating Systems [Spring 2025] School of Engineering and Technology, University of Washington - Tacoma	L18.22
--------------	---	--------

22

LATENCY TIMES

- Design considerations:
  - SSDs 4x the time of DRAM
  - HDDs 80x the time of DRAM

Action	Latency (ns)	(μs)	
L1 cache reference	0.5ns		
L2 cache reference	7 ns		14x L1 cache
Mutex lock/unlock	25 ns		
Main memory reference	100 ns		20x L2 cache, 200x L1
Read 4K randomly from SSD*	150,000 ns	150 μs	~1GB/sec SSD
Read 1 MB sequentially from memory	250,000 ns	250 μs	
Read 1 MB sequentially from SSD*	1,000,000 ns	1,000 μs	1 ms ~1GB/sec SSD, 4X memory
Read 1 MB sequentially from disk	20,000,000 ns	20,000 μs	20 ms 80x memory, 20X SSD

- Latency numbers every programmer should know
- From: <https://gist.github.com/jboner/2841832#file-latency-txt>

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.23

23

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - **Swapping Mechanisms** Swapping Policies
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.24

24

SWAP SPACE

- Disk space for storing memory pages
- “Swap” them in and out of memory to disk as needed

	PFN 0	PFN 1	PFN 2	PFN 3
Physical Memory	Proc 0 [VPN 0]	Proc 1 [VPN 2]	Proc 1 [VPN 3]	Proc 2 [VPN 0]

	Block 0	Block 1	Block 2	Block 3	Block 4	Block 5	Block 6	Block 7
Swap Space	Proc 0 [VPN 1]	Proc 0 [VPN 2]	[Free]	Proc 1 [VPN 0]	Proc 1 [VPN 1]	Proc 3 [VPN 0]	Proc 2 [VPN 1]	Proc 3 [VPN 1]

Physical Memory and Swap Space

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.25

25

SWAP SPACE - 2

- The size of the swap space can be seen using the Linux free command: “free -h”

```
wlloyd@dlone:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	30G	11G	14G	1.3G	4.4G	17G
Swap:	31G	0B	31G			

- With sufficient disk space, a common allocation is to create Swap space greater than or equal to physical RAM

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.26

26

SWAP SPACE - 3

- Swap space lives on a separate logical volume in Ubuntu Linux that is managed separately from the root file system
- Check logical volumes with “sudo lvsdisplay” command:

```
--- Logical volume ---
LV Path                /dev/ubuntu-vg/swap_1
LV Name                 swap_1
VG Name                ubuntu-vg
LV UUID                G1ovj6-4M33-2YXY-YETH-wF7V-93vF-QRQytG
LV Write Access         read/write
LV Creation host, time ubuntu, 2018-09-30 15:44:16 -0700
LV Status               available
# open                  2
LV Size                 976.00 MiB
Current LE              244
Segments                1
Allocation              inherit
Read ahead sectors      auto
- currently set to     256
Block device            253:1
```

- See also “lvm lvs” command

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.27

27

PAGE LOCATION

- Memory pages are:
  - Stored in memory
  - Swapped to disk
- Present bit
  - In the page table entry (PTE) indicates if page is present
- Page fault
  - Memory page is accessed, but has been swapped to disk

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.28

28

PAGE FAULT

- OS steps in to handle the page fault
- Loading page from disk requires a free memory page
- Page-Fault Algorithm

```
1:      PFN = FindFreePhysicalPage()  
2:      if (PFN == -1)                // no free page found  
3:          PFN = EvictPage()         // run replacement algorithm  
4:      DiskRead(PTE.DiskAddr, pfn)   // sleep (waiting for I/O)  
5:      PTE.present = True            // set PTE bit to present  
6:      PTE.PFN = PFN                 // reference new loaded page  
7:      RetryInstruction()            // retry instruction
```

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.29

29

PAGE REPLACEMENTS

- Page daemon
  - Background threads which monitors swapped pages
- Low watermark (LW)
  - Threshold for when to swap pages to disk
  - Daemon checks: free pages < LW
  - Begin swapping to disk until reaching the highwater mark
- High watermark (HW)
  - Target threshold of free memory pages
  - Daemon free until: free pages >= HW

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.30

30

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, **Swapping Policies**
- Ch. 36 I/O Devices, Ch. 37 Hard Disk Drives


June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.31

31

REPLACEMENT  
POLICIES



June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.3  
2

32

CACHE MANAGEMENT

- Replacement policies apply to “any” cache
- Goal is to minimize the number of misses
- Average memory access time (**AMAT**) can be estimated:

$$AMAT = (P_{Hit} * T_M) + (P_{Miss} * T_D)$$

Argument	Meaning
$T_M$	The cost of accessing memory (time)
$T_D$	The cost of accessing disk (time)
$P_{Hit}$	The probability of finding the data item in the cache(a hit)
$P_{Miss}$	The probability of not finding the data in the cache(a miss)

- Consider  $T_M = 100\text{ ns}$ ,  $T_D = 10\text{ms}$
- Consider  $P_{hit} = .9\text{ (90\%)}$ ,  $P_{miss} = .1$
- Consider  $P_{hit} = .999\text{ (99.9\%)}$ ,  $P_{miss} = .001$

June 3, 2025

TCCS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.33

33

OPTIMAL REPLACEMENT POLICY

- What if:
  - We could predict the future (... with a magical oracle)
  - All future page accesses are known
  - Always replace the page in the cache used farthest in the future
- Used for a comparison
- Provides a “best case” replacement policy
- Consider a 3-element empty cache with the following page accesses:

0 1 2 0 1 3 0 3 1 2 1

What is the hit/miss ratio?

6 hits

June 3, 2025

TCCS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.34

34

FIFO REPLACEMENT

- Queue based
- **Always replace the oldest element** at the back of cache
- Simple to implement
- Doesn't consider importance... just arrival ordering
- Consider a 3-element empty cache with the following page accesses:

0 1 2 0 1 3 0 3 1 2 1

- What is the hit/miss ratio?
- How is FIFO different than LRU?

4 hits

LRU incorporates history

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.35

35

RANDOM REPLACEMENT

- Pick a page at random to replace
- Simple and fast implementation
- Performance depends on luck of random choices

0 1 2 0 1 3 0 3 1 2 1

Number of Hits	Frequency
1	0
2	2
3	10
4	20
5	40
6	45

Random Performance over 10,000 Trials

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.36

36

HISTORY-BASED POLICIES

- LRU: Least recently used
- Always replace page with oldest access time (front)
- Always move end of cache when element is read again
- LRU requires constant reorganization of the cache
- Considers temporal locality (*when pg was last accessed*)

0 1 2 0 1 3 0 3 1 2 1

What is the hit/miss ratio?

6 hits

- LFU: Least frequently used
- Always replace page with the fewest # of accesses (front)
- Incorporates frequency of use - *must track pg accesses*
- Consider frequency of page accesses

0 1 2 0 1 3 0 3 1 2 1

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.37

37

LFU

- LFU: Least frequently used
- Always replace page with the fewest # of accesses (front)
- Incorporates frequency of use - *must track pg accesses*
- Consider frequency of page accesses

0 1 2 0 1 3 0 3 1 2 1

What is the hit/miss ratio?

Hit/miss ratio is=6 hits

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.38

38

2 hits

3 hits

4 hits

5 hits

6 hits

Consider a 3-element cache. With a FIFO replacement policy, how many hits occur with the following page access sequence:  
1 2 0 1 3 1 2 0 2 1 3

June 3, 2025

TCSS422: Operating Systems [Spring 2025]

L18.9

39

2 hits

3 hits

4 hits

5 hits

6 hits

Consider a 3-element cache. With an LRU replacement policy, how many hits occur with the following page access sequence:  
1 2 0 1 3 1 2 0 2 1 3

June 3, 2025

TCSS422: Operating Systems [Spring 2025]

L18.0

40

WORKLOAD EXAMPLES: NO-LOCALITY

No-Locality (Random Access) Workload

Perform 10,000 random page accesses

Across set of 100 memory pages

The No-Locality Workload

Cache Size (Blocks)	OPT Hit Rate	LRU Hit Rate	FIFO Hit Rate	RAND Hit Rate
0	0%	0%	0%	0%
20	20%	20%	20%	20%
40	40%	40%	40%	40%
60	60%	60%	60%	60%
80	80%	80%	80%	80%
100	100%	100%	100%	100%

When the cache is large enough to fit the entire workload, it doesn't matter which policy you use.

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.41

41

WORKLOAD EXAMPLES: 80/20

80/20 Workload

Perform 10,000 page accesses, against set of 100 pages

80% of accesses are to 20% of pages (hot pages)

20% of accesses are to 80% of pages (cold pages)

The 80-20 Workload

Cache Size (Blocks)	OPT Hit Rate	LRU Hit Rate	FIFO Hit Rate	RAND Hit Rate
0	0%	0%	0%	0%
20	80%	70%	40%	20%
40	90%	85%	60%	40%
60	95%	92%	75%	60%
80	98%	96%	85%	75%
100	100%	100%	100%	100%

LRU is more likely to hold onto hot pages  
(recalls history)

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.42

42

Slides by Wes J. Lloyd

L18.21

# WORKLOAD EXAMPLES: SEQUENTIAL

- **Looping sequential workload**
  - Refer to 50 pages in sequence: 0, 1, ..., 49
  - Repeat loop

The Looping-Sequential Workload

Cache Size (Blocks)	OPT Hit Rate (%)	LRU Hit Rate (%)	FIFO Hit Rate (%)	RAND Hit Rate (%)
0	0	0	0	0
20	40	0	0	10
40	80	0	0	40
50	100	0	0	100
60	100	0	0	100
80	100	0	0	100
100	100	100	100	100

Random performs better than FIFO and LRU for cache sizes < 50

Algorithms should provide "scan resistance"

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.43

43

## With small cache sizes, for the looping sequential workload, why do FIFO and LRU fail to provide cache hits?

Cache hits in this scenario require consideration of how frequently accessed memory is for cache replacement

Memory accesses are unpredictable and too random. Unpredictable accesses require a random cache replacement policy for cache hits

Memory accesses to elements that are accessed repeatedly are too spread apart temporally to benefit from caching

Unlike Random cache replacement, both FIFO and LRU fail to speculate memory accesses in advance to improve caching

None of the above

Start the presentation to see live content. For screen share software, share the entire screen. Get help at [pollev.com/app](https://pollev.com/app)

44

## IMPLEMENTING LRU

- Implementing last recently used (LRU) requires tracking access time for all system memory pages
- Times can be tracked with a list
- For cache eviction, we must scan an entire list
- Consider: 4GB memory system ( $2^{32}$ ), with 4KB pages ( $2^{12}$ )
- This requires  $2^{20}$  comparisons !!!
- Simplification is needed
  - Consider how to approximate the oldest page access

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.45

45

## IMPLEMENTING LRU - 2

- Harness the Page Table Entry (PTE) Use Bit
- HW sets to 1 when page is used
- OS sets to 0
- Clock algorithm (*approximate LRU*)
  - Refer to pages in a circular list
  - Clock hand points to current page
  - Loops around
    - IF USE\_BIT=1 set to USE\_BIT = 0
    - IF USE\_BIT=0 replace page



June 3, 2025

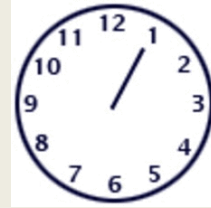
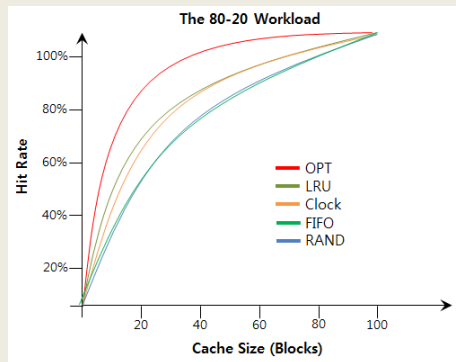
TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.46

46

## CLOCK ALGORITHM

- Not as efficient as LRU, but better than other replacement algorithms that do not consider history



June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.47

47

## CLOCK ALGORITHM - 2

- Consider dirty pages in cache
- If DIRTY (modified) bit is FALSE
  - No cost to evict page from cache
- If DIRTY (modified) bit is TRUE
  - Cache eviction requires updating memory
  - Contents have changed
- Clock algorithm should favor no cost eviction

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.48

48

WHEN TO LOAD PAGES

- On demand → demand paging
- Prefetching
  - Preload pages based on anticipated demand
  - Prediction based on locality
  - Access page P, suggest page P+1 may be used
- What other techniques might help anticipate required memory pages?
  - Prediction models, historical analysis
  - In general: accuracy vs. effort tradeoff
  - High analysis techniques struggle to respond in real time

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.49

49

OTHER SWAPPING POLICIES

- Page swaps / writes
  - Group/cluster pages together
  - Collect pending writes, perform as batch
  - Grouping disk writes helps amortize latency costs
- Thrashing
  - Occurs when system runs many memory intensive processes and is low in memory
  - Everything is constantly swapped to-and-from disk

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.50

50

## OTHER SWAPPING POLICIES - 2

- Working sets
  - Groups of related processes
  - When thrashing: prevent one or more working set(s) from running
  - Temporarily reduces memory burden
  - Allows some processes to run, reduces thrashing

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.51

51

## OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices Ch. 37 Hard Disk Drives

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma


L18.52

52

# CHAPTER 36: I/O DEVICES

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma



53

## OBJECTIVES

- Chapter 36
  - I/O: Polling vs Interrupts
  - Programmed I/O (PIO)
    - Port-mapped I/O (PMIO)
    - Memory-mapped I/O (MMIO)
  - Direct memory Access (DMA)

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma


L18.54

54


# I/O DEVICES

Modern computer systems interact with a variety of devices

## input



## output



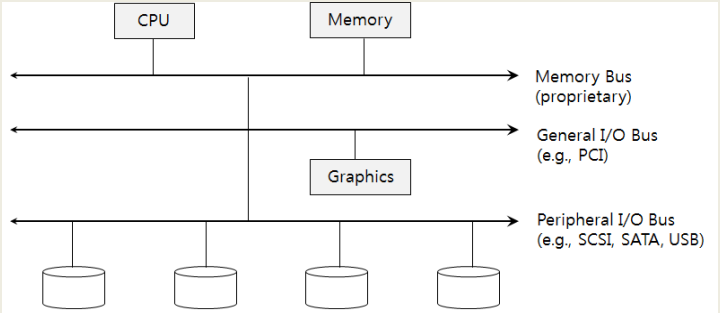
June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.55

55

# COMPUTER SYSTEM ARCHITECTURE



**Prototypical System Architecture**

**VERY FAST:** CPU is attached to main memory via a Memory bus.

**FAST:** High speed devices (e.g. video) are connected via a General I/O bus.

**SLOWER:** Disks are connected via a Peripheral I/O bus.

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.56

56

I/O BUSES

- Buses
  - Buses closer to the CPU are faster
  - Can support fewer devices
  - Further buses are slower, but support more devices
- Physics and costs dictate “levels”
  - Memory bus
  - General I/O bus
  - Peripheral I/O bus
- Tradeoff space: speed vs. locality

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.57

57

CANONICAL DEVICE

- Consider an arbitrary canonical “*standard/generic*” device

Registers:    Status    Command    Data

Micro-controller(CPU)  
Memory (DRAM or SRAM or both)  
Other Hardware-specific Chips

interface

internals

Canonical Device

- Two primary components
  - Interface (registers for communication)
  - Internals: Local CPU, memory, specific chips, firmware (embedded software)

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.58

58

CANONICAL DEVICE:  
HARDWARE INTERFACE

- Status register
  - Maintains current device status
- Command register
  - Where commands for interaction are sent
- Data register
  - Used to send and receive data to the device

General concept:  
The OS interacts and controls device behavior  
by reading and writing the device registers.

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.59

59

OS DEVICE INTERACTION

- Common example of device interaction

```
while ( STATUS == BUSY) ← Poll- Is device available?  
; //wait until device is not busy  
write data to data register ← Command parameterization  
write command to command register ← Send command  
    Doing so starts the device and executes the command  
while ( STATUS == BUSY) ← Poll – Is device done?  
; //wait until device is done with your request
```

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.60

60

POLLING

- OS checks if device is *READY* by repeatedly checking the **STATUS** register
  - Simple approach
  - CPU cycles are wasted without doing meaningful work
  - Ok if only a few cycles, for rapid devices that are often **READY**
  - BUT** polling, as with “spin locks” we understand is inefficient

CPU utilization by polling

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.61

61

INTERRUPTS VS POLLING

- For longer waits, put process waiting on I/O to sleep
- Context switch (C/S) to another process
- When I/O completes, fire an interrupt to initiate C/S back
  - Advantage: better multi-tasking and CPU utilization
  - Avoids: unproductive CPU cycles (polling)

Diagram of CPU utilization by interrupt

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.62

62

## INTERRUPTS VS POLLING - 2

### What is the tradeoff space ?

- Interrupts are not always the best solution

- How long does the device I/O require?
- What is the cost of context switching?

If device I/O is fast → polling is better.  
When I/O time < 1 CPU time slice (e.g. 10 ms)

If device I/O is slow → interrupts are better.  
When I/O time > 1 CPU time slice

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.63

63

## INTERRUPTS VS POLLING - 3

- Alternative: two-phase hybrid approach

- Initially poll, then sleep and use interrupts

- Issue: livelock problem

- Common with network I/O
- Many arriving packets generate **many many** interrupts
- Overloads the CPU!
- No time to execute code, just interrupt handlers !

- Livelock optimization

- Coalesce multiple arriving packets (for different processes) into fewer interrupts
- Must consider number of interrupts a device could generate

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.64

64

## DEVICE I/O

- To interact with a device we must send/receive DATA
- There are two general approaches:
  - Programmed I/O (PIO):
    - Port mapped I/O (PMIO)
    - Memory mapped I/O (MMIO)
  - Direct memory access (DMA)

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.65

65

Transfer Modes			
Mode ↕	# ↕	Maximum transfer rate (MB/s) ↕	cycle time ↕
PIO	0	3.3	600 ns
	1	5.2	383 ns
	2	8.3	240 ns
	3	11.1	180 ns
	4	16.7	120 ns
Single-word DMA	0	2.1	960 ns
	1	4.2	480 ns
	2	8.3	240 ns
Multi-word DMA	0	4.2	480 ns
	1	13.3	150 ns
	2	16.7	120 ns
	3 <sup>[34]</sup>	20	100 ns
	4 <sup>[34]</sup>	25	80 ns
Ultra DMA	0	16.7	240 ns + 2
	1	25.0	160 ns + 2
	2 (Ultra ATA/33)	33.3	120 ns + 2
	3	44.4	90 ns + 2
	4 (Ultra ATA/66)	66.7	60 ns + 2
	5 (Ultra ATA/100)	100	40 ns + 2
	6 (Ultra ATA/133)	133	30 ns + 2
7 (Ultra ATA/167) <sup>[35]</sup>	167	24 ns + 2	

From <https://en.wikipedia.org/wiki/Parallel ATA>

From <https://en.wikipedia.org/wiki/Parallel ATA>

66

## PROGRAMMED I/O (PIO)

- I/O performed on the CPU
- CPU time is consumed performing I/O
- CPU supports data movement (input/output)
- PIO is slow: CPU is occupied with meaningless work

**PIO**

“over-burdened”

1 : task 1

2 : task 2

C : copy data from memory

CPU

1	1	1	1	C	C	C	2	2	2	2	2	1	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Disk

1	1	1	1	1
---	---	---	---	---

Diagram of CPU utilization

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.67

67

## PIO DEVICES

- Legacy serial ports
- Legacy parallel ports
- PS/2 keyboard and mouse
- Legacy MIDI, joysticks
- Old network interfaces

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.68

68

## PROGRAMMED I/O DEVICE (PIO) INTERACTION

- Two primary PIO methods
  - Port mapped I/O (PMIO)
  - Memory mapped I/O (MMIO)

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.69

69

## PORT MAPPED I/O (PMIO)

- Device specific CPU I/O Instructions
- Follows a CISC model:  
specific CPU instructions used for device I/O
  - x86-x86-64: `in` and `out` instructions
  - `outb`, `outw`, `outl`
  - 1, 2, 4 byte copy from EAX → device's I/O port

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.70

70

MEMORY MAPPED I/O (MMIO)

- Device’s memory is mapped to standard memory addresses
- MMIO is common with RISC CPUs:  
Special CPU instructions for PIO eliminated
- Old days: 16-bit CPUs didn’t have a lot of spare memory space
- Today’s CPUs have LARGE address spaces:  
32-bit (4GB addr space) & 64-bit (128 TB addr space)
- Device I/O uses regular CPU instructions usually used to read/write memory to access device
- Device is mapped to unique memory address **reserved** for I/O
  - Address must not be available for normal memory operations.
  - Generally very high addresses (out of range of type addresses)
- Device monitors CPU address bus and respond to instructions on their addresses

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.71

71

DIRECT MEMORY ACCESS (DMA)

- Copy data in memory by **offloading** to “DMA controller”
- Many devices (including CPUs) integrate DMA controllers
- CPU gives DMA: memory address, size, and copy instruction
- DMA performs I/O independent of the CPU
- DMA controller generates CPU interrupt when I/O completes

The diagram illustrates the interaction between a CPU, a DMA controller, and a disk. At the top, a legend indicates that '1' represents task 1, '2' represents task 2, and 'C' represents copy data from memory. The CPU row shows a sequence of tasks: four '1's, followed by eight '2's, and then three '1's. The DMA row shows three 'C's, which occur during the execution of task 2. The Disk row shows five '1's, which occur while the DMA controller is copying data. This demonstrates that the DMA controller can move data between memory and the disk without the CPU being involved in the transfer itself.

Diagram of CPU utilization by DMA

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.72

72

## DIRECTORY MEMORY ACCESS – 2

- Many devices use DMA
  - HDD/SSD controllers (ISA/PCI)
  - Graphics cards
  - Network cards
  - Sound cards
  - Intra-chip memory transfer for multi-core processors
- DMA allows computation and data transfer time to proceed in parallel

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.73

73

## DEVICE INTERACTION

- The OS must interact with a variety of devices
- Example: Consider a file system that works across a variety of types of disks:
  - SCSI, IDE, USB flash drive, DVD, etc.
- File system should be general purpose, where device specific I/O implementation details are abstracted
- **Device drivers** use abstraction to provide general interfaces for vendor specific hardware
- In Linux: block devices

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.74

74

FILE SYSTEM ABSTRACTION

- Layers of I/O abstraction in Linux
- C functions (open, read, write) issue **block read and write** requests to the generic block layer

The diagram illustrates the File System Stack, divided into user and kernel spaces by a dashed line. In the user space, the Application layer contains the POSIX API (open, read, write, close, etc). In the kernel space, the stack consists of the File System layer (containing the Generic Block Interface for block read/write), the Generic Block Layer (containing the Specific Block Interface for protocol-specific read/write), and the Device Driver layer (supporting SCSI, ATA, etc).

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.75

75

FILE SYSTEM ABSTRACTION ISSUES

- Too much abstraction**
  - Many devices provide special capabilities
  - Example: SCSI Error handling
  - SCSI devices provide extra details which are lost to the OS
- Buggy device drivers**
  - 70% of OS code is in device drivers
  - Device drivers are required for every device plugged in
  - Drivers are often 3<sup>rd</sup> party, which is not quality controlled at the same level as the OS (Linux, Windows, MacOS, etc.)

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.76

76

OBJECTIVES – 6/3

- Questions from 5/29
- Assignment 2 - June 5 AOE
- Assignment 3 (as a Tutorial) - June 10 AOE
- Memory Segmentation Activity + answers (available in Canvas)
- Quiz 4 – Page Tables - Due June 8 AOE
- Final exam – Thursday June 12 @ 3:40pm
- Tutorial 3 - File Systems (Optional, Extra Credit)
- Chapter 21/22: Beyond Physical Memory
  - Swapping Mechanisms, Swapping Policies
- Ch. 36 I/O Devices, **Ch. 37 Hard Disk Drives**


June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.77

77

CH. 37:  
HARD DISK DRIVES



June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.78

78

OBJECTIVES

- Chapter 37
  - HDD Internals
    - Seek time
    - Rotational latency
    - Transfer speed
    - Capacity
    - Scheduling algorithms

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.79

79

HARD DISK DRIVE (HDD)

- Primary means of data storage (persistence) for decades
  - Remains inexpensive for high capacity storage
  - 2020: 16 TB HDD - \$400, ~15.3 TB SSD - \$4,380
- Consists of a large number of data **sectors**
- Sector size is 512-bytes
- An n sector HDD  
can be is addressed as an array of 0..n-1 sectors

June 3, 2025

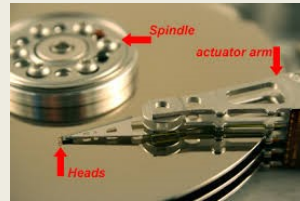
TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.80

80

## HDD INTERFACE

- Writing disk sectors is atomic (512 bytes)
- Sector writes are completely successful, or fail
- Many file systems will read/write 4KB at a time
  - Linux ext3/4 default filesystem blocksize – 4096
- Same as typical memory page size



June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.81

81

## BLOCK SIZE IN LINUX EXT4

- `mkefs.ext4 -i <bytes-per-inode>`
- Formats disk w/ ext4 fileys with specified byte-to-inode ratio
- Today's disks are so large, some use cases with many small files can run out of inodes before running out of disk space
- Each inode record tracks a file on the disk
- Larger bytes-per-inode ratio results in fewer inodes
  - Default is around ~4096
- Value shouldn't be smaller than blocksize of filesystem
- Note: It is not possible to expand the number of inodes after the filesystem is created, - be careful deciding the value
- Check inode stats: `tune2fs -l /dev/sda1` (← disk dev name)

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.82

82

EXAMPLE: USDA SOIL EROSION MODEL  
WEB SERVICE (RUSLE2)

- Host ~2,000,000 small XML files totaling 9.5 GB on a ~20GB filesystem on a cloud-based Virtual Machine
- With default inode ratio (4096 block size), only ~488,000 files will fit
- Drive less than half full, but files will not fit !
- HDDs support a minimum block size of 512 bytes
- OS filesystems such as ext3/ext4 can support “finer grained” management at the expense of a larger catalog size
  - Small inode ratio- inodes will considerable % of disk space

June 3, 2025

TCCS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.83

83

EXAMPLE: USDA SOIL EROSION MODEL  
WEB SERVICE (RUSLE2) - 2

- Free space in bytes (df)

Device	total size	bytes-used	bytes-free	usage
/dev/vda2	13315844	9556412	3049188	76% /mnt

- Free inodes (df -i) @ 512 bytes / node

Device	total inodes	used	free	usage
/dev/vda2	3552528	1999823	1552705	57% /mnt

June 3, 2025

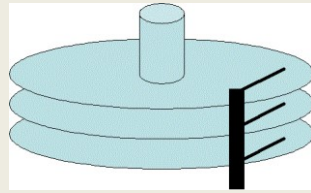
TCCS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.84

84

## HDD INTERFACE - 2

- **Torn write**
  - When OS uses larger block size than HDD
  - Block writes not **atomic** - they SPAN multiple HDD sectors
  - Upon power failure only a portion of the OS block is written – *can lead to data corruption...*
- **HDD access**
  - Sequential reads of sectors is fastest
  - Random sector reads are slow
  - Disk head continuously must jump to different tracks



June 3, 2025

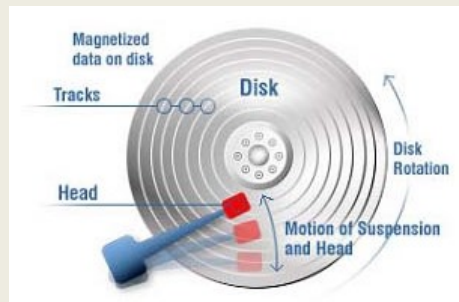
TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.85

85

## HDD PLATTER

- Made from aluminum coated with thin magnetic layer
- HDD records on both sides of each platter
- Data is stored by inducing magnetic changes



June 3, 2025

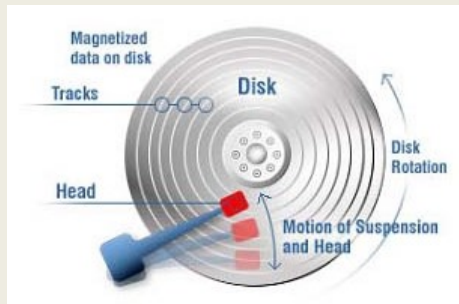
TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.86

86

## HDD SPINDLE

- Connected to motor which spins the disk
- Speed measures in RPM (rotations per minute)
- Typical: 7200-15000 rpm
- 10000 rpm – 1 rotation in 6ms; 15k rpm 1 rotation in 4ms



June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

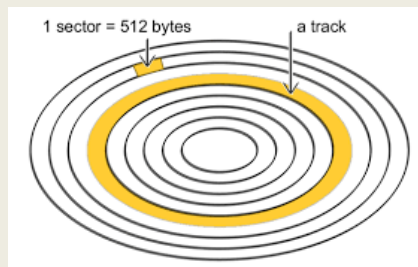
L18.87

87

## HDD TRACK

- Concentric circle of sectors
- Single side of platter contains 290 K tracks (2008)
- Zones: groups of tracks with same # of sectors

**Outer tracks have  
More sectors**



June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.88

88

EXAMPLE: SIMPLE DISK DRIVE

- Single track disk
- Head: one per surface of drive
- Arm: moves heads across surface of platters

A diagram of a single track disk. A central spindle is shown with a circular disk around it. The disk is divided into 12 numbered segments (0-11) representing a single track. An arm is attached to the spindle, with a head positioned over segment 6. An arrow above the disk indicates counter-clockwise rotation. The text 'Rotates this way' is above the arrow. Below the diagram is the caption 'A Single Track Plus A Head'.

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.89

89

HARD DISK STRUCTURE

A diagram of a hard disk structure. A central spindle is shown with multiple platters stacked on it. Each platter is divided into sectors. An arm assembly is shown with multiple read-write heads, one for each platter. The diagram is labeled with 'track t', 'sector s', 'cylinder c', 'platter', 'spindle', 'arm assembly', 'read-write head', 'arm', and 'rotation'.

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.90

90

SINGLE-TRACK LATENCY:  
THE ROTATIONAL DELAY

- **Rotational latency** ( $T_{\text{rotation}}$ ): time to rotate to desired sector
- Average  $T_{\text{rotation}}$  is ~ about half the time of a full rotation
- How to calculate  $T_{\text{rotation}}$  from rpm
  1. Calculate time for 1 rotation based on rpm
    - > Convert rpm to rps
  2. Divide by two (*average rotational latency*)
- 7200rpm = 8.33ms per rotation /2= ~4.166ms
- 10000rpm = 6ms per rotation /2= ~3ms
- 15000rpm = 4ms per rotation /2= ~2ms

Rotates this way

head

arm

spindle

A Single Track Plus A Head

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.91

91

SEEK TIME

Rotates this way

Rotates this way

Remaining rotation

seek

spindle

Three Tracks Plus A Head (Right: With Seek)  
(e.g., read to sector 11)

- **Seek time** ( $T_{\text{seek}}$ ): time to move disk arm to proper track
- Most time consuming HDD operation

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.92

92

FOUR PHASES OF SEEK

- Acceleration → coasting → deceleration →settling
- **Acceleration**: the arm gets moving
- **Coasting**: arm moving at full speed
- **Deceleration**: arm slow down
- **Settling**: Head is carefully positioned over track
  - Settling time is often high, from .5 to 2ms

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.93

93

HDD I/O

- Data transfer
  - Final phase of I/O: time to read or write to disk surface
- Complete I/O cycle:
  1. Seek (accelerate, coast, decelerate, settle)
  2. Wait on rotational latency (*until track aligns*)
  3. Data transfer

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.94

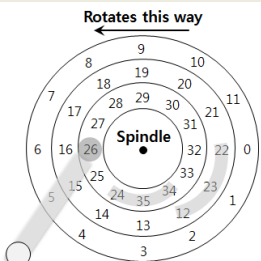
94

TRACK SKEW

Sectors are offset across tracks to allow time for head to reposition for sequential reads

Without track skew, when head is repositioned sector would have already been passed

Rotates this way



Three Tracks: Track Skew Of 2

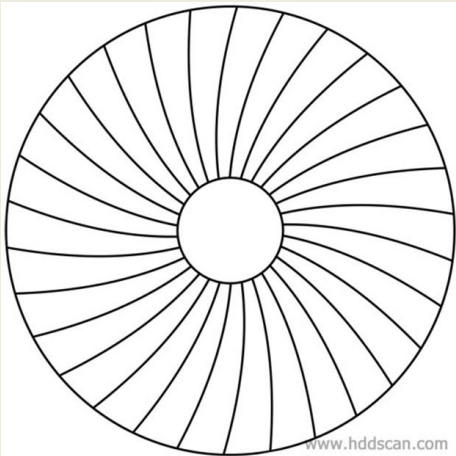
June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.95

95

TRACK SKEW - 2



www.hddscan.com

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.96

96

Slides by Wes J. Lloyd

L18.48

## HDD CACHE

- Buffer to support caching reads and writes
- Improves drive response time
- Up to 256 MB, slowly have been growing
- Two styles
  - Writeback cache
    - Report write complete immediately when data is transferred to HDD cache
    - Dangerous if power is lost
  - Writethrough cache
    - Reports write complete only when write is physically completed on disk

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.97

97

## TRANSFER SPEED

- Can calculate I/O transfer speed with:

- I/O Time:  $T_{I/O} = T_{seek} + T_{rotation} + T_{transfer}$

- $T_{transfer} = \text{DATA}_{size} \times \text{Rate}_{I/O}$

- Rate of I/O:  $R_{I/O} = \frac{Size_{transfer}}{T_{I/O}}$

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.98

98

EXAMPLE: I/O SPEED

■ Compare two disks:

1. Random workload: 4KB (random read on HDD)

2. Sequential workload: 100MB (contiguous sectors)

> Calculate  $T_{\text{rotation}}$  from rpm (rpm→rps, time for 1 rotation / 2)

	Cheetah 15K.5	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Average Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	16/32 MB
Connects Via	SCSI	SATA

Disk Drive Specs: SCSI Versus SATA

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.99

99

EXAMPLE: I/O SPEED

1. Random workload: 4KB (random read on HDD)

2. Sequential workload: 100MB (contiguous sectors)

$T_{I/O} = T_{\text{seek}} + T_{\text{rotation}} + T_{\text{transfer}}$

$T_{\text{transfer}} = \text{Data}_{\text{size}} \times \text{Rate}_{I/O}$

$R_{I/O} = \frac{\text{Size}_{\text{transfer}}}{T_{I/O}}$

		Cheetah 15K.5	Barracuda
4 KB Random	$T_{\text{seek}}$	4 ms	9 ms
	$T_{\text{rotation}}$	2 ms	4.2 ms
	$T_{\text{transfer}}$	30 microseconds	38 microseconds
100 MB Sequential	$T_{I/O}$	6 ms	13.2 ms
	$R_{I/O}$	0.66 MB/s	0.31 MB/s
	$T_{\text{transfer}}$	800 ms	950 ms
	$T_{I/O}$	806 ms	963.2 ms
	$R_{I/O}$	125 MB/s	105 MB/s

Disk Drive Performance: SCSI Versus SATA

There is a huge gap in drive throughput between random and sequential workloads

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.100

100

MODERN HDD SPECS

- See sample HDD configurations here:
  - Up to 20 TB
- <https://www.westerndigital.com/products/data-center-drives#hard-disk-hdd>

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.101

101

DISK SCHEDULING

- Disk scheduler: determine how to order I/O requests
- Multiple levels - OS and HW
- OS: provides ordering
- HW: further optimizes using intricate details of physical HDD implementation and state

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.102

102

SSTF – SHORTEST SEEK TIME FIRST

- Disk scheduling – which I/O request to schedule next
- Shortest Seek Time First (SSTF)
- Order queue of I/O requests by nearest track

Rotates this way

SSTF: Scheduling Request 21 and 2  
Issue the request to 21 → issue the request to 2

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.103

103

SSTF ISSUES

- Problem 1: HDD abstraction
  - Drive geometry not available to OS. Nearest-block-first is a comparable alternate algorithm.
- Problem 2: Starvation
  - Steady stream of requests for local tracks may prevent arm from traversing to other side of platter

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.104

104

DISK SCHEDULING ALGORITHMS

- **SCAN (SWEEP)**
  - Perform single repeated passes back and forth across disk
  - Issue: if request arrives for a recently visited track it will not be revisited until a full cycle completes
- **F-SCAN**
  - Freeze incoming requests by adding to queue during scan
  - Cache arriving requests until later
  - Delays help avoid starvation by postponing servicing nearby newly arriving requests vs. requests at edge of sweep
  - Provides better fairness
- **Elevator (C-SCAN)** – circular scan
  - Sweep only one direction (e.g. outer to inner) and repeat
  - SCAN favors middle tracks vs. outer tracks with 2-way sweep

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.105

105

SHORTEST TIME POSITIONING FIRST

- Determine next sector to read?
  - Where:  $T_{seek} = T_{rotation}$
- On which track?
- On which sector?

SSTF: Sometimes Not Good Enough

On modern drives, both seek and rotation are roughly equivalent:  
Thus, SPTF (Shortest Positioning Time First) is useful.

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.106

106

## OPTIMIZATION: I/O MERGING

- Group temporary adjacent requests
- Reduce overhead
- Read (memory blocks): 33 8 34
- How long we should wait for I/O ?
- When do we know we have waited too long?

June 3, 2025

TCSS422: Operating Systems [Spring 2025]  
School of Engineering and Technology, University of Washington - Tacoma

L18.107

107

## QUESTIONS



108